Predictive Lossless Coding Of Images And Video

Technical Field

The invention relates to lossless coding of images and video.

Background

Lossless image coding has a large variety of important applications, including high quality digital photography, filmography, graphics, etc. It also applies to professional grade video coding, for encoding video frames at the highest possible quality setting, i.e., losslessly. Images in these applications can have diverse characteristics, which presents a difficult challenge for designing an image codec to be generically applicable across these applications. For example, images in graphics typically have sharp edges or transitions in color (e.g., between text and background colors, and at borders of adjacent shapes), whereas photographic images generally are continuous tone (i.e., vary continuously in color (e.g., as a gradient) across the image).

Due to differences in image characteristics, most generic image codecs are either designed to compress photographic (e.g. JPEG) or graphic images (GIF). Photographic image compression usually uses a de-correlating transform like DCT or wavelets, whereas graphic image compression typically uses string based codecs such as LZ77 or LZ78. In general, photographic codecs don't work well on graphics because the basic assumption of local smoothness or DC-bias which underlies transform methods is usually broken in graphics. Conversely, graphics codecs do poorly on photographic images because the alphabet is too large to build a good dictionary. As a result, existing image codecs for photographic images are not designed for easy interoperability with image and video codecs, nor do they handle graphics content efficiently.

For example, CALIC [as described by X. Wu, N. Memon and K. Sayood, "A Context-Based Adaptive Lossless/Nearly-Lossless Coding Scheme For Continuous-Tone Images," ISO, 1995], JPEG-LS [as described by M.J. Weinberger and G. Seroussi, "The

LOCO-I Lossless Image Compression Algorithm: Principles and Standardization into JPEG-LS," *IEEE Trans. Image Processing*, Vol. 9, pp. 1309-1324, August 2000] and SPIHT [as described by Said and W.A. Pearlman, "A New Fast And Efficient Image Codec Based On Set Partitioning In Hierarchical Trees," *IEEE Trans. On Circuits and Systems for Video Technology*, Vol. 6, No. 6, pp. 243-250, June 1996] are current state-of-art lossless image codecs for photographic images. However, they are not designed for easy interoperability with image and video codecs, nor do they handle graphics content efficiently. On the other hand, GIF is a current state-of-art lossless graphics codec. But, it too doesn't handle photographic content, nor is it easy to incorporate inside a video codec. PTC [as described by H.S. Malvar, "Fast Progressive Image Coding Without Wavelets," pp. 243-252, DCC 2000] is a macroblock-based codec that can be easily integrated into image and/or video codecs. However, it does not do very well with graphics content. Further, BTPC [described by J.A. Robinson, "Efficient General-Purpose Image Compression With Binary Tree Predictive Coding," *IEEE Transactions on Image Processing*, Vol. 6, No. 4, April 1997] is designed to handle photographic and graphic images in a unified and speed optimized design, but its compression is far from adequate.

Summary

Predictive lossless coding (PLC) addresses the problem discussed above of providing lossless image compression that is generically applicable to a wide variety of images and video. A PLC implementation illustrated herein provides lossless image compression of a wide variety of photographic content (image, video and graphic alike) with compression efficiency comparable to existing lossless image codecs, and performing at faster run-time complexity than most lossless image codecs.

Various implementations of predictive lossless coding described herein achieve these results through a combination of at least some of the following points:

1 Operates on the YCoCg color space. This color space improves coding efficiency of photographic and graphic image content, and further a color space conversion

to/from the prevalent RGB (red-green-blue) color representation can be performed using a fast, all integer procedure.

2 Operates on macroblocks. This ensures that the predictive lossless coding can be easily integrated into existing image and video codecs, and is conducive to scalable space/time implementations in hardware and software, e.g., using slices.

3 Uses a rich set of local, differential pulse-code modulation (DPCM) predictions at the macroblock level. These are designed to optimally decorrelate image data from photographic as well as graphics sources, without resorting to abnormal samplings or re-scanning of the image (cf., J.A. Robinson, "Efficient General-Purpose Image Compression With Binary Tree Predictive Coding," *IEEE Transactions on Image Processing*, Vol. 6, No. 4, April 1997; and X. Wu, N. Memon and K. Sayood, "A Context-Based Adaptive Lossless/Nearly-Lossless Coding Scheme For Continuous-Tone Images," ISO, 1995). The DPCM's are also designed to produce residuals that have zero biased, two-sided Laplacian distributions, as these are best coded by the run-length Golomb Rice (RLGR) entropy coding method.

4 Uses the run length Golomb Rice (RLGR) method to entropy code the various symbol distributions.

In one implementation employing this combination of points, the predictive lossless coding provides lossless compression of all photographic content (image, video and graphic alike) which is not only equal that of CALIC and greater than other existing formats including JPEG-LS, PTC, BTPC, etc., but also results in twice or greater compression for most graphic content compared to these photographic-type codecs, and offers run-time complexity which is faster by twice or greater than the most advanced state-of-the-art photographic-type codecs.

Additional features and advantages of the invention will be made apparent from the following detailed description of embodiments that proceeds with reference to the accompanying drawings.

Brief Description Of The Drawings

Figure 1 is a block diagram of an image encoder utilizing predictive lossless coding.

Figure 2 is a diagram depicting a macroblock and slice structure of the predictive lossless coding utilized in the encoder of Figure 1.

Figure 3 is a diagram showing the neighborhood for a pixel in the macroblock on which the DPCM prediction in the encoder of Figure 1 is based.

Figure 4 is a diagram depicts DPCM prediction modes employed in the predictive lossless coding utilized in the encoder of Figure 1.

Figure 5 is a pseudo-code listing of the encoding procedure for the predictive lossless coding utilized in the encoder of Figure 1.

Figure 6 is a block diagram of an image decoder utilizing predictive lossless coding.

Figure 7 is a block diagram of a suitable computing environment for implementing the PLC codec of Figures 1 and 6.

Detailed Description

The following description is directed to implementations of predictive lossless coding that combines a mix of some or all of run-length Golomb Rice (RLGR) entropy encoding, multiple DPCM modes, the YCoCg color space and a macroblock (MB) coding structure to provide an efficient and fast codec applicable to a wide variety of image content, including photographic (continuous tone), graphic, and video.

1. PLC Encoder

With reference now to Figure 1, an illustrative example of an image encoder 100 that is based on predictive lossless coding (PLC) performs encoding or compression of image data 105. The image data that is input to the PLC encoder can be in any of various uncompressed image data formats. For example, a common format processed by the illustrated image encoder is red-green-blue (RGB) image data, such as for a photographic

or graphic image, a frame of video, etc. This RBG image data is generally structured as a two dimensional array of picture elements (pixels), where each pixel is represented as a red-green-blue (RGB) color sample of the image. Alternative implementations of the image encoder can use other input image data formats. It should further be recognized that this encoder can be incorporated within a video encoder for encoding a frame within a video sequence, using the predictive lossless coding.

The PLC image encoder 100 processes this image data through a set of processes, which include a color space converter 110, a macro-block splitter 120, a DPCM modulator 130, and a RLGR entropy encoder 140. The color space converter 110 converts pixels of the input image data from a displayable color space representation into the YCoCg color space, which improves coding efficiency. The macro-block splitter 120 splits the image into macro-blocks, for compatibility with macro-block and slice based image and video codecs. The DPCM encoder 130 selects and applies one of a set of available DPCM prediction modes to each individual macro-block that produces prediction residuals having a distribution suitable for RGR entropy coding. The RLGR entropy encoder 140 then encodes the prediction residuals of the macro-block. This produces a PLC encoded representation of the image data.

1.1 YCoCg Color Space Converter

More specifically, the color space converter 110 converts the color format of the input image data to the YCoCg color space. The pixels of the input image data are typically represented in a readily-displayable color space format, such as the red-green-blue (RGB) color space. The YCoCg color space is more suitable for space-efficient image coding. The YCoCg has been found to work well both for photographic and graphic images, and outperforms other color transforms in terms of coding gain. More specifically, the YCoCg lossless color space gave an improvement of ~15% as compared to the RGB color space in the PLC encoding.

The RGB to YCoCg color space conversion is done prior to any encoding of the RGB-format input image data. In this illustrated implementation of the PLC encoder 100, the color space converter 110 uses the RGB to YCoCg conversion process described

in more detail in H.S. Malvar and G.J. Sullivan, "YCoCg-R: A Color Space With RGB Reversibility and Low Dynamic Range," Joint Video Team of ISO/IEC MPEG & ITU-T VCEG Doc. JVT-1014, July, 2003, which provides a way to losslessly de-correlate RGB into YCoCg color space. The RGB to YCoCg color space conversion can be effected by the forward transform defined in the following equation (1).

$$\begin{bmatrix} Y \\ Co \\ Cg \end{bmatrix} = \begin{bmatrix} \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \\ \frac{1}{2} & 0 & -\frac{1}{2} \\ -\frac{1}{4} & \frac{1}{2} & -\frac{1}{4} \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{1}$$

The decoder 600 (Figure 6) may include a conversion back to the RGB color space. This conversion uses the inverse operation defined in equation (2).

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 & 1 & -1 \\ 1 & 0 & 1 \\ 1 & -1 & -1 \end{bmatrix} \begin{bmatrix} Y \\ Co \\ Cg \end{bmatrix} \tag{2}$$

Lifting steps are used to obtain both forward and backward transforms, resulting in fast all integer conversion procedures. The lifting steps in the forward direction are given by –

Co = R - B;
x = B + (Co / 2);
Cg = G - x;
Y = x + (Cg / 2);

And in the inverse direction by –

x = Y – (Cg / 2);
G = x + Cg;
B = x - (Co / 2);
R = Co + B;

As can be seen, these lifting steps can be implemented with all integer adds, subtracts and bit shifts, all of which are extremely fast

1.2 Macro block coding

The macro-block splitter 120 splits the image into macro blocks (MBs), as depicted in Figure 2. In one implementation of the image encoder 100, each MB is 16 x 16 pixels in size. Alternative implementations can use other sizes of macro-blocks. This macro-block structure makes it practical and easy to plug the PLC-based image encoder into popular image as well as video codecs. It also enables hardware and software implementations to easily use slice coding, in which the encoded bit stream is packetized into slices. The slices are typically some integer number of rows of MBs. This makes for a flexible memory footprint, and easily lends itself to space/time scalability (e.g., by using parallel or multi-threaded execution units).

The macro-block splitter splits each plane of the YCoCg color space into MBs, and encodes them separately. With each individual MB, the PLC image encoder 100 encodes the following syntax elements: a DPCM mode, a MB mode and the DPCM residuals.

The DPCM mode element identifies the DPCM mode chosen by the DPCM modulator 130 to decorrelate data in this MB. In this implementation, the DPCM modulator chooses from eight possible DPCM modes, although fewer or more DPCM modes may be used in alternative PLC encoder implementations. The macro-block splitter uses a separate RLGR context to encode the DPCM mode for the MB.

For the MB mode element, the macro-block splitter uses a two symbol alphabet, with a separate dedicated RLGR coding context. The MB mode signals one of the following two events - a) The MB encodes DPCM values; or b) The MB is “flat”, and is therefore skipped. In the latter case, the event is treated as an early exit from encoding/decoding the MB’s pixels as DPCM residuals, as the DPCM mode element is sufficient to re-generate all the pixel values. This flat macro-block mode coding is described in more detail below.

Finally, if the MB is not skipped, the DPCM residuals are encoded in the MB segment of the PLC-encoded output stream.

1.3 DPCM Modes

The DPCM modulator 130 chooses and applies a DPCM mode for the current MB that more optimally decorrelates the MB to produce DPCM residuals that compress better with RLGR entropy coding. The RLGR entropy coding achieves its best coding performance when its input values have a zero-biased, two-sided Laplacian distribution. For example, the DPCM modulator 130 in the illustrated PLC-based encoder can switch between eight different DPCM modes to decorrelate each MB. Alternative implementations can include fewer or more DPCM modes. These various different DPCM modes are designed to produce residuals having this optimal distribution for various different MB pixel patterns. This allows different dominant edge directions that may fall inside the MB to be coded efficiently.

With reference to Figure 3, the various DPCM modes specify which of a pixel's neighbors 300 are used to predict the pixel's value. More specifically, the value of each pixel 310 of the macro block is predicted from some combination of one or more neighboring pixels 320-323. The difference obtained from subtracting a pixel's actual value from its predicted value is the DPCM residual value of that pixel. The eight DPCM modes in the illustrated DPCM modulator 130 use predictions based on combinations of the neighboring left pixel 320, top-left pixel 321, top pixel 322, and top-right pixel 323. This allows the DPCM mode to be applied in a single, one-way pass or scan through the MB (i.e., scanning each row of pixels left-to-right, from top to bottom rows of the MB).

Figure 4 depicts the eight DPCM prediction modes used in the illustrated DPCM modulator 130. These modes are designed to decorrelate various common pixel patterns into a set of residuals having a symbol distribution that is better suited to RLGR entropy coding. More specifically, these DPCM prediction modes are as follows:

Mode 0: This is the "raw" or no DPCM mode, where individual pixels are encoded directly without any subtraction. This mode is useful for random or "speckle" type MBs with no consistently good predictor throughout the MB.

Mode 1: In this DPCM mode, the pixel value is subtracted from its immediate left neighbor prior to encoding. This mode is useful when the major edges lie along the horizontal.

Mode 2: In this DPCM mode, the pixel value is subtracted from its immediate top neighbor. This mode is useful when the major edges lie along the vertical.

Mode 3: In this DPCM mode, the value is either subtracted from the minimum of its left and top neighbors, or alternatively from the maximum of its left and top neighbors. This DPCM mode is useful for ramp diagonal edges that pass through the current pixel position.

Mode 4: In this DPCM mode, the value is subtracted from the average of its top and top right neighbors. This DPCM mode is useful for diagonal ramp edges with a different orientation.

Mode 5: In this DPCM mode, the value is subtracted from its top-left neighbor. This DPCM mode is useful for diagonal bands, e.g., in graphic content.

Mode 6: In this DPCM mode, the modulator 130 subtracts the same value from its left neighbor as the difference between its top and top-left neighbors. This mode is useful for banded horizontal ramps.

Mode 7: In this DPCM mode, the modulator 130 subtracts the average of the left and top neighbors. This is also useful when diagonal edges dominate in the MB.

The DPCM modulator tests each of the DPCM prediction modes 1 through 7 (i.e., other than the no DPCM mode, which is mode 0) so as to choose which DPCM mode produces more compressible DPCM residuals. The DPCM modulator applies the respective DPCM modes and checks the symbol distribution of the resulting residuals. The DPCM modulator then checks which prediction mode produced residuals having a distribution closest to the ideal distribution for RLGR entropy encoding. The DPCM modulator further checks whether this closest distribution is sufficiently close to the ideal zero-biased, two-sided Laplacian distribution. The DPCM modulator chooses the DPCM prediction mode with the closest-to-ideal distribution for the macro block, unless the sufficiency threshold is not met. Otherwise, if the DPCM prediction mode with the closest-to-ideal distribution does not meet the sufficiency threshold, then the DPCM modulator chooses the no DPCM mode (mode 0) as a default.

1.4 Flat MB Mode Coding

As discussed above, the PLC-based encoder 100 also can encode a MB in the flat MB mode. The flat MB mode is used when the DPCM residuals resulting from application of a DPCM mode to the MB are all zero. When testing the DPCM modes to choose the DPCM mode for use on the MB, the DPCM modulator further checks whether the currently tested DPCM mode produces all zero DPCM residuals for the MB. Upon determining that a DPCM mode produces all zero DPCM residuals, the PLC-based encoder 100 then encodes the MB in the flat MB mode – without needing to test further DPCM modes. When encoding in the flat MB mode, the PLC-based encoder 100 can encode the MB in the output bitstream as only the MB mode and the DPCM mode (i.e., skipping encoding the DPCM residuals). The coding of the flat MB mode and the DPCM mode for the MB in the output bitstream are sufficient to decode the values of the MB's pixels. Because the DPCM residuals are not encoded, the flat MB mode yields greater compression efficiency for encoding the MB.

1.5 Multiple Run Length Golomb Rice (RLGR) Contexts

With reference again to Figure 1, the MB mode, DPCM mode and DPCM residuals produced by the DPCM modulator 130 are then entropy encoded using RLGR coding in the RLGR entropy encoder 140. The RLGR entropy encoder in the illustrated PLC-based encoder 100 uses the run-length Golomb-Rice coding process described in H. Malvar, "Fast Progressive wavelet coding," *Proc. IEEE Data Compression Conference*, Snowbird, UT, pp. 336-343, March-April 1999. This RLGR coding process is not a truly generic entropy coder, such as an adaptive arithmetic coder. The RLGR coding makes the assumption that the most probable symbol is zero. So, if a string of numbers with the most probable symbol not being zero is fed to RLGR, the RLGR coding will have poor coding performance. If its input comes from a source with nearly Laplacian symbol distribution, the RLGR coding process will encode that data very well, quite closely to the entropy, and in many cases it will do a better job than adaptive arithmetic encoding. In the PLC-based encoder 100, the DPCM prediction modes are designed to produce zero-biased two-sided Laplacian distributions of signed integers (for both photographic

and graphic images) of common photographic and graphic image content, on which RLGR works best.

The RLGR entropy encoder 140 in the illustrated PLC-based encoder uses a separate RLGR context for each of: a) the MB mode (flat or not); b) the DPCM mode; c) the DPCM residual values (zero biased two-sided Laplacian distributions of integers). In each of these contexts, the RLGR entropy encoder performs an adaptive run-length/Golomb-Rice binary encoding of the binary string formed by the significant bits that come from the symbols being coded by the separate context, e.g., the DPCM residuals for the DPCM residual values context. The use of multiple RLGR contexts to code different symbol distributions improves the entropy coding performance. This is because it is very important to adapt to each individual distribution and its idiosyncratic skew. As an example, the MB modes are more likely to be all not flat. Accordingly, their distribution is likely to be skewed away from the flat case. But, this could turn around if the content was graphic rather than photographic. Dedicating a specific RLGR context allows the RLGR entropy encoder in the PLC-base encoder 100 to adapt to such unimodal distributions with greater efficiency. In alternative implementations of the PLC-based encoder, more or fewer RLGR contexts can be used. The use of additional separate RLGR contexts to encode the DPCM residuals in such implementations could provide greater entropy coding gains and prevent context dilution, but three separate contexts are used in the illustrated implementation for practicality.

After entropy encoding by the RLGR entropy encoder, the bitstream multiplexor 150 assembles the RLGR encoded data for the MB into an output bitstream 195. In implementations using slice coding, the bitstream multiplexor assembles or packetizes the encoded MBs into slices.

1.6 Pseudo-Code Listing

The PLC coding performed in the above-described PLC-based encoder 100 is summarized in the pseudo-code listing 500 shown in Figure 5. In this pseudo-code listing, the "ImageBand" input parameter represents image data from one of the color

space co-ordinates, i.e. Y, Co or Cg. This PLC coding process is invoked after the color space conversion of the image to the YCoCg color space.

2. PLC Decoder

With reference now to Figure 6, an image decoder 600 based on predictive lossless coding (PLC) performs decoding of the output bitstream 195 produced from PLC-coding by the PLC-based image encoder 100. In this PLC-based image decoder 600, a bitstream demultiplexer 610 first separates out individual encoded MBs in the bitstream, and the encoded MB mode, the DPCM mode and DPCM residuals for that MB. The bistream demultiplexer provides the separate data to an RLGR decoder 620.

The RLGR decoder 620 decodes the RLGR-encoded MB mode, DPCM mode and DPCM residuals for each MB. The RLGR decoder 620 uses the RLGR decoding process as described in H. Malvar, "Fast Progressive wavelet coding," *Proc. IEEE Data Compression Conference*, Snowbird, UT, pp. 336-343, March-April 1999. The RLGR decoder 620 then provides the decoded data to a DPCM demodulator 630.

The DPCM demodulator 630 performs the inverse process on the DPCM residuals for the DPCM prediction mode that was used for the MB, thus restoring the MB data. For a MB encoded in flat MB mode, the DPCM demodulator 630 performs the inverse process for all zero residuals for the decoded DPCM prediction mode.

After the inverse DPCM prediction is applied, an image reconstructor 640 re-assembles the MBs to reconstruct the image. A color space converter 650 then performs the inverse of the YCoCg color space conversion to convert this image data back into an RGB image. In some implementations, this conversion can be omitted, and the image left in the YCoCg color space format.

3. Computing Environment

The above described PLC-based encoder 100 and/or decoder 600 (PLC codec) can be implemented on any of a variety of image and video processing devices and computing devices, including computers of various form factors (personal, workstation, server, handheld, laptop, tablet, or other mobile), distributed computing networks and Web services, and image and video recorders/players/receivers/viewers, as a few general

examples. The PLC-based codec can be implemented in hardware circuitry, as well as in codec software 780 executing within a computer or other computing environment, such as shown in Figure 7.

Figure 7 illustrates a generalized example of a suitable image/video processing device in a computing environment 700 (e.g., of a computer) in which the described techniques can be implemented. This environment 700 is not intended to suggest any limitation as to scope of use or functionality of the invention, as the present invention may be implemented in diverse general-purpose or special-purpose image/video processing environments.

With reference to Figure 7, the computing environment 700 includes at least one processing unit 710 and memory 720. In Figure 7, this most basic configuration 730 is included within a dashed line. The processing unit 710 executes computer-executable instructions and may be a real or a virtual processor. In a multi-processing system, multiple processing units execute computer-executable instructions to increase processing power. The memory 720 may be volatile memory (e.g., registers, cache, RAM), non-volatile memory (e.g., ROM, EEPROM, flash memory, etc.), or some combination of the two. The memory 720 stores software 780 implementing the PLC-based codec.

A computing environment may have additional features. For example, the computing environment 700 includes storage 740, one or more input devices 750, one or more output devices 760, and one or more communication connections 770. An interconnection mechanism (not shown) such as a bus, controller, or network interconnects the components of the computing environment 700. Typically, operating system software (not shown) provides an operating environment for other software executing in the computing environment 700, and coordinates activities of the components of the computing environment 700.

The storage 740 may be removable or non-removable, and includes magnetic disks, magnetic tapes or cassettes, CD-ROMs, CD-RWs, DVDs, or any other medium which can be used to store information and which can be accessed within the computing environment 700. The storage 740 stores instructions for the PLC-based codec software 780.

The input device(s) 750 (e.g., for devices operating as a control point in the device connectivity architecture 100) may be a touch input device such as a keyboard, mouse, pen, or trackball, a voice input device, a scanning device, or another device that provides input to the computing environment 700. For audio, the input device(s) 750 may be a sound card or similar device that accepts audio input in analog or digital form, or a CD-ROM reader that provides audio samples to the computing environment. The output device(s) 760 may be a display, printer, speaker, CD-writer, or another device that provides output from the computing environment 700.

The communication connection(s) 770 enable communication over a communication medium to another computing entity. The communication medium conveys information such as computer-executable instructions, audio/video or other media information, or other data in a modulated data signal. A modulated data signal is a signal that has one or more of its characteristics set or changed in such a manner as to encode information in the signal. By way of example, and not limitation, communication media include wired or wireless techniques implemented with an electrical, optical, RF, infrared, acoustic, or other carrier.

The macro expansion processing and display techniques herein can be described in the general context of computer-readable media. Computer-readable media are any available media that can be accessed within a computing environment. By way of example, and not limitation, with the computing environment 700, computer-readable media include memory 720, storage 740, communication media, and combinations of any of the above.

The techniques herein can be described in the general context of computer-executable instructions, such as those included in program modules, being executed in a computing environment on a target real or virtual processor. Generally, program modules include routines, programs, libraries, objects, classes, components, data structures, etc. that perform particular tasks or implement particular abstract data types. The functionality of the program modules may be combined or split between program modules as desired in various embodiments. Computer-executable instructions for program modules may be executed within a local or distributed computing environment.

For the sake of presentation, the detailed description uses terms like "determine," "generate," "adjust," and "apply" to describe computer operations in a computing environment. These terms are high-level abstractions for operations performed by a computer, and should not be confused with acts performed by a human being. The actual computer operations corresponding to these terms vary depending on implementation.

In view of the many possible embodiments to which the principles of our invention may be applied, we claim as our invention all such embodiments as may come within the scope and spirit of the following claims and equivalents thereto.